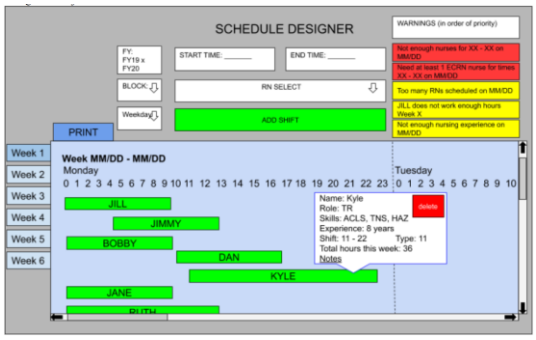
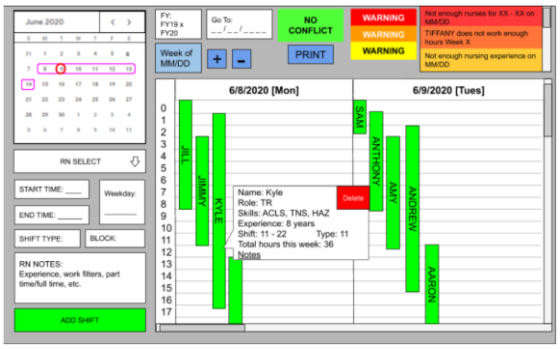
Objective 2 - Front-end

2.1 Create UI components

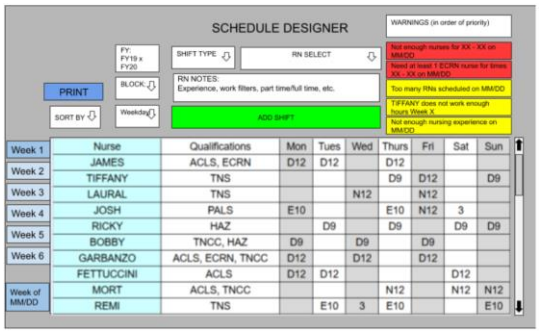
Rationale: Create UI components which can be re-used in the future as needed. Based on user interviews conducted, key interface interactions and functionality needs were identified. Initial mock-ups of front-end user schedule assignment interface were completed. Four models were designed for review – see Figure 14.



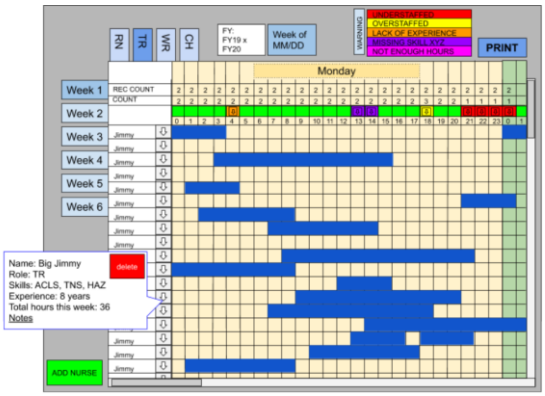
Design #1: Adjusted GANTT Chart Schedule Planner



Design #3: Vertical Schedule Planner



Design #2: Spreadsheet Style Schedule Planner



Design #4: Spreadsheet Style Alt. Schedule Plan

Figure 14: Alternative Design Mockups for the Planning Module

The mock-ups in Figure 14 were then reviewed and evaluated for feedback by the internal Medecipher team as well as external potential users and expert emergency medicine nursing and physician subject matter experts. Example team feedback in Figure 15. Preferred features from each design have been combined into specs for the planned front-end design.

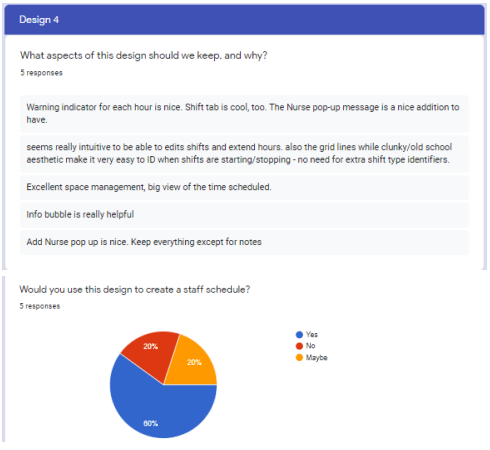


Figure 15: Example team feedback from initial UI mockups

Technical tasks: Develop the ability to make certain staff/provider recommendation choices with the help of the user interface. We will develop frontend components such as the Schedule View (Figure 16 and 17) which allows the user to choose a date and recommendation type (conservative, moderate, or extreme). This then sends a request to the REST API

which will pulls back and caches data for the request, returning data points, and a schedule which displays recommendations for all three recommendation types. We will develop a feedback loop to record user input.

Expected results and alternative approaches: unit tested client-agnostic, reusable AngularJS Component. Design is an ongoing process, and we have learned that each component within the web page commands its own aesthetic design.

2.2 API/Creation of a service layer

Rationale: This layer will be responsible for abstracting out the API communication. Depending on the environment the service layer will either call real backend or return mock data to the Gantt chart component design in Objective 2.1

Technical tasks: We created the technology platform and built out several of the components of the service layer in Phase I proof-of-concept. In Phase II, we will make the following changes: the API will now be part of the Kubernetes/Micro Services container architecture. Additionally, the data science engine will be containerized for scalability.

2.3 Wire service layer and UI components

Rationale: The service layer and the UI components are wired together to pass data between one another.

Technical tasks: In Phase II, we will stand up a Kubernetes/Microservices environment (see Section 3.1). We will wire this new service layer to our UI via our existing API developed in Phase I. Figure 18 demonstrates how connectivity will be achieved in Phase II.

Expected results and alternative approaches: UI component can communicate with the backend API via service layer.

2.4 Development of integration & component testing

Rationale: Develop requirements documentation to test the newly developed features against the requirements. Both service layer and UI components are unit tested and end-to-end tested.

Technical tasks: We will create a continuous delivery pipeline for automated testing. During the build and deployment process, specific unit tests may be conducted using API calls and evaluating expected results. End-to-end tests will be performed the same way, but interacting as a user, rather than direct API calls.

Figure 16: Schedule View – Shift

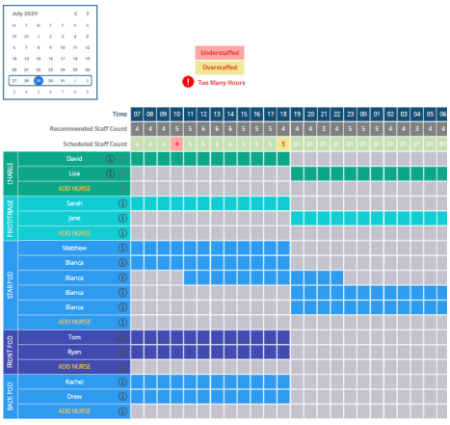


Figure 17: Schedule View - Count of nurses per area/hour

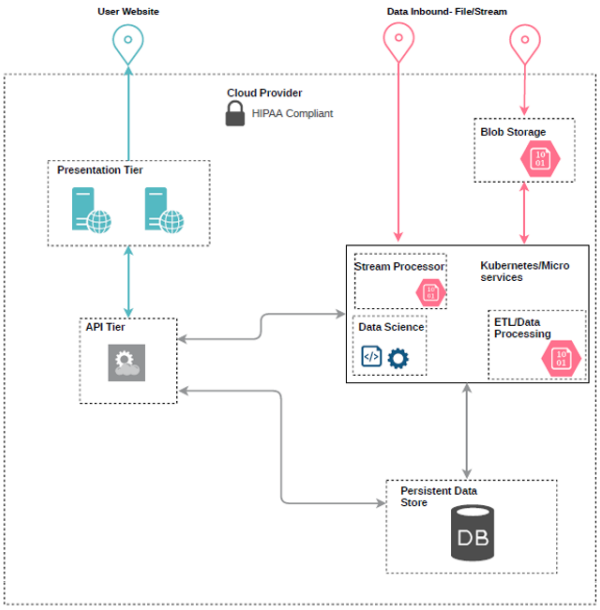


Figure 18: Overall System Architecture – Phase II